

---

## Parallel Processing in Combustion Analysis

Greg Schunk  
NASA/MSFC  
University of Alabama in Huntsville  
richard.schunk@msfc.nasa.gov

T. J. Chung  
University of Alabama in Huntsville

# Introduction

---

- The objective of this research is to demonstrate the application of the Flow-field Dependent Variation (FDV) method to a problem of current interest in supersonic chemical combustion.
- Due in part to the stiffness of the chemical reactions, the solution of such problems on unstructured three dimensional grids often dictates the use of parallel computers.
- Preliminary results for the injection of a supersonic hydrogen stream into vitiated air are presented.

## Flow-field Dependent Variation Approach

---

- The conservation of mass for a chemical species,  $k$ , may be represented as follows:

$$\frac{\partial U_k}{\partial t} = B_k - \frac{\partial F_i}{\partial x_i} - \frac{\partial G_i}{\partial x_i}$$

$$U_k = [\rho Y_k]$$

$$F_i = [\rho Y_k v_i]$$

$$G_i = [-\rho D_{km} Y_{k,i}]$$

$$B_k = w_k$$

where  $Y_k$  represents the species mass fraction,  $F$  is the convective flux,  $G$  is the diffusive flux, and  $w_k$  is the generation of species  $k$  from chemical reaction.

- The change in the mass of species  $k$  over a single timestep,  $\Delta t$ , may be determined from the second order mixed explicit/implicit formulation shown below where the first and second order variational parameters,  $s_1$  and  $s_2$ , are introduced to control the degree of implicit damping:

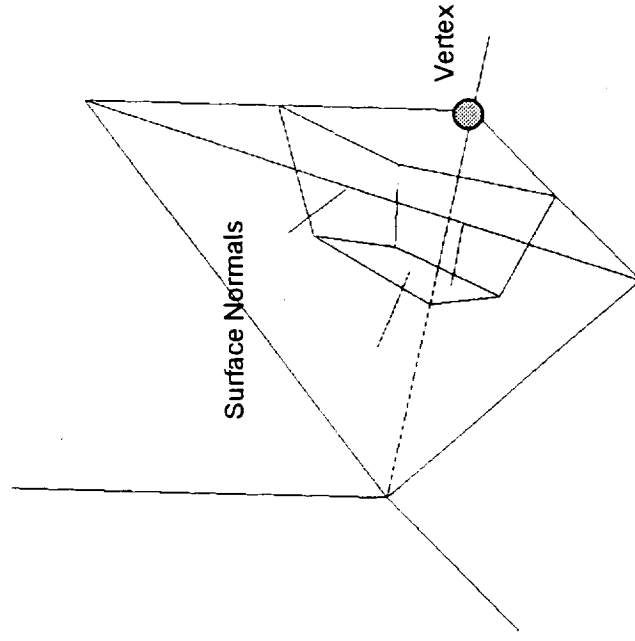
$$\Delta U_k^{N+1} - s_1 \Delta t \frac{\partial \Delta U_k^{N+1}}{\partial t} + s_2 \frac{\Delta t^2}{2} \frac{\partial^2 \Delta U_k^{N+1}}{\partial t^2} = \Delta t \frac{\partial U_k^N}{\partial t} + \frac{\Delta t^2}{2} \frac{\partial^2 U_k^N}{\partial t^2}$$

$s_1, s_2 \rightarrow 0$  Fully explicit

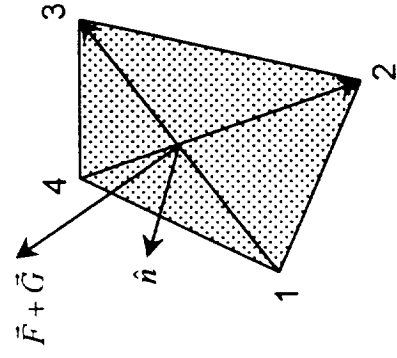
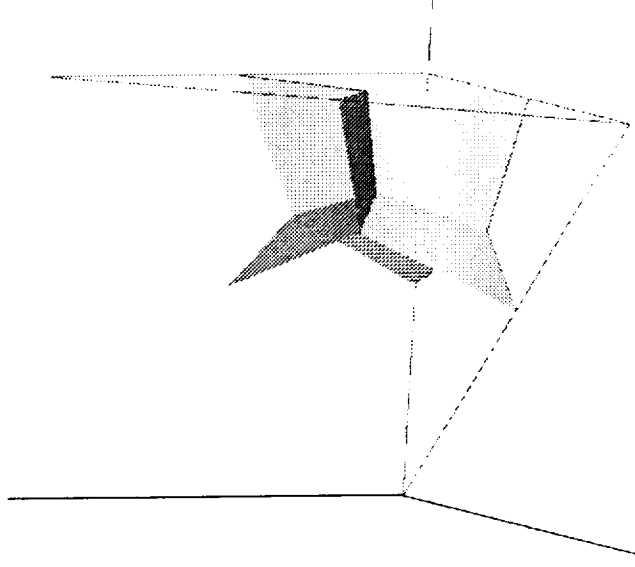
$s_1, s_2 \rightarrow 1$  Fully implicit

# Control Volume Finite Element Method

Vertex Control Volume



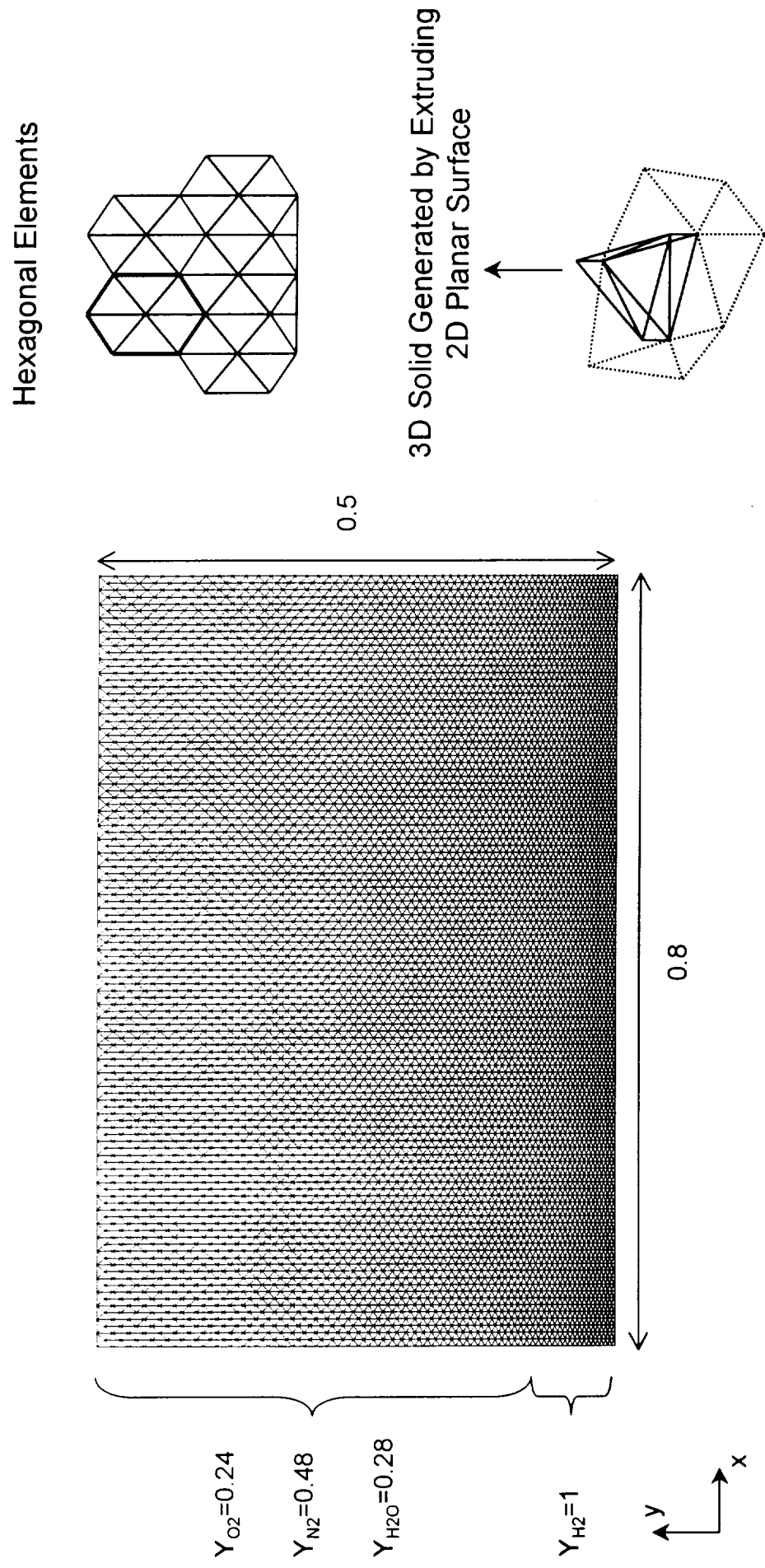
Discretization of Tetrahedral Volume



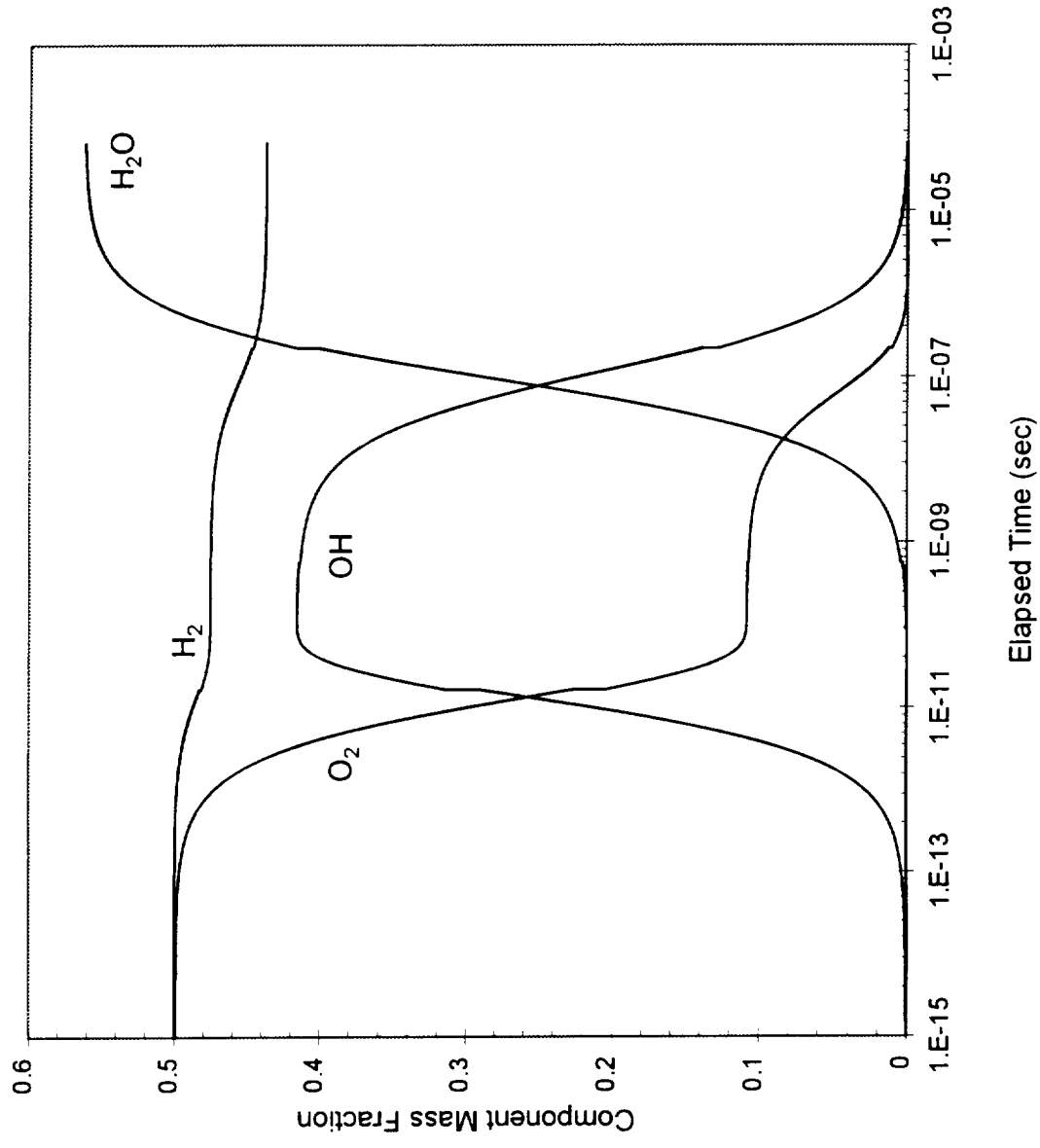
$$\frac{\partial U}{\partial t} = B - \vec{\nabla} \cdot (\vec{F} + \vec{G})$$

$$\int \left[ \frac{\partial U}{\partial t} - B \right] dV = - \int \vec{\nabla} \cdot (\vec{F} + \vec{G}) dV = - \int (F_i + G_i) n_i d\Gamma$$

# Computational Grid

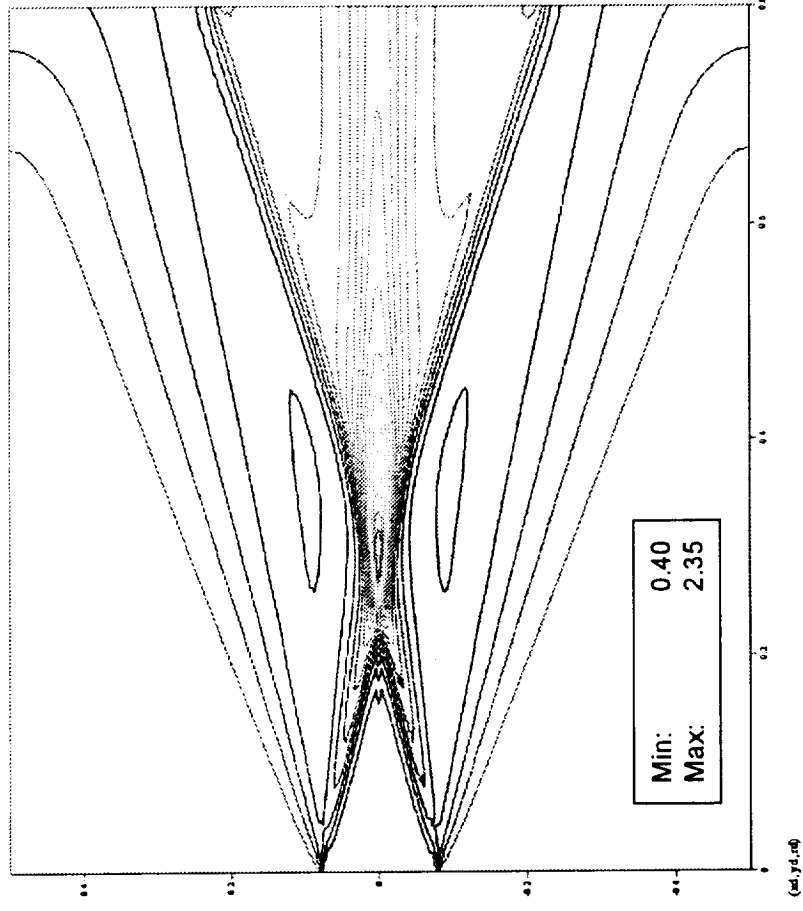


# Reaction of Premixed Hydrogen and Oxygen using Two Step Global Combustion Model

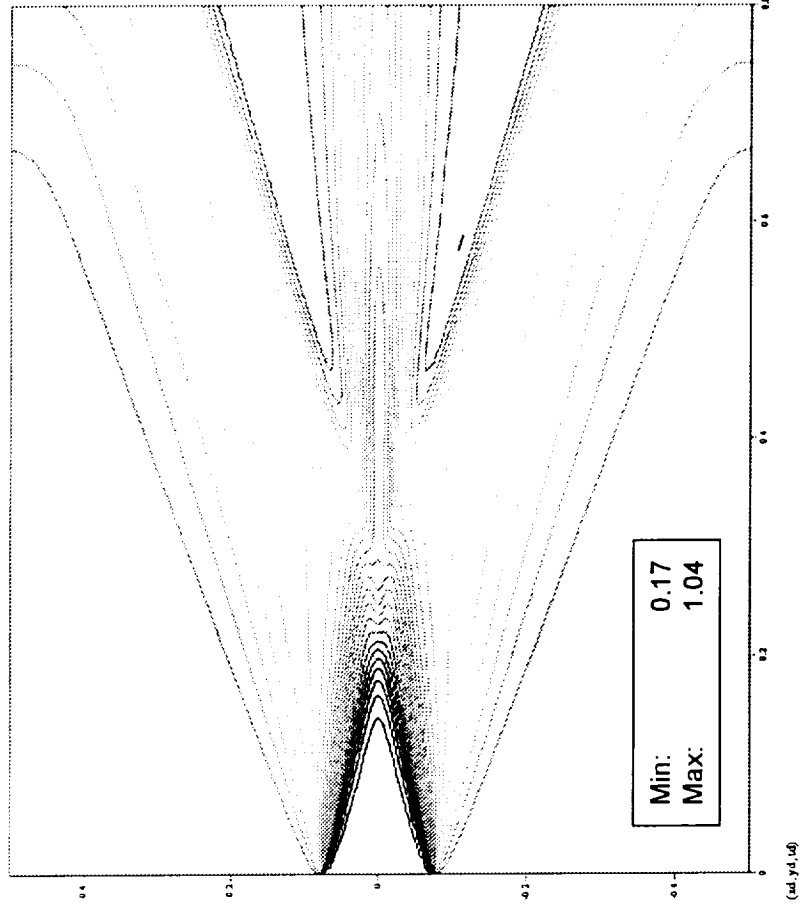


# Density and Temperature Contours for Non-reacting Flow-field

Density



Temperature





## Parallel Programming: Processes and Threads

---

- When a computational task is delivered to the operating system (OS) of a computer for execution, the OS responds by creating *a process*.
  - The OS allocates memory for the process, provides access to system resources, and schedules time for the process to run.
  - Processes do not normally share resources, but may communicate through mechanisms provided by the OS.
- Within a process, there exists data and program segments and the execution path through the program segment may be thought of as *a thread*.
- If sections of the program may be executed concurrently, then multiple threads through the process may be created.
- Like processes, threads are scheduled for execution by the OS, but share global memory (within the same process) alleviating the need for process level communication.

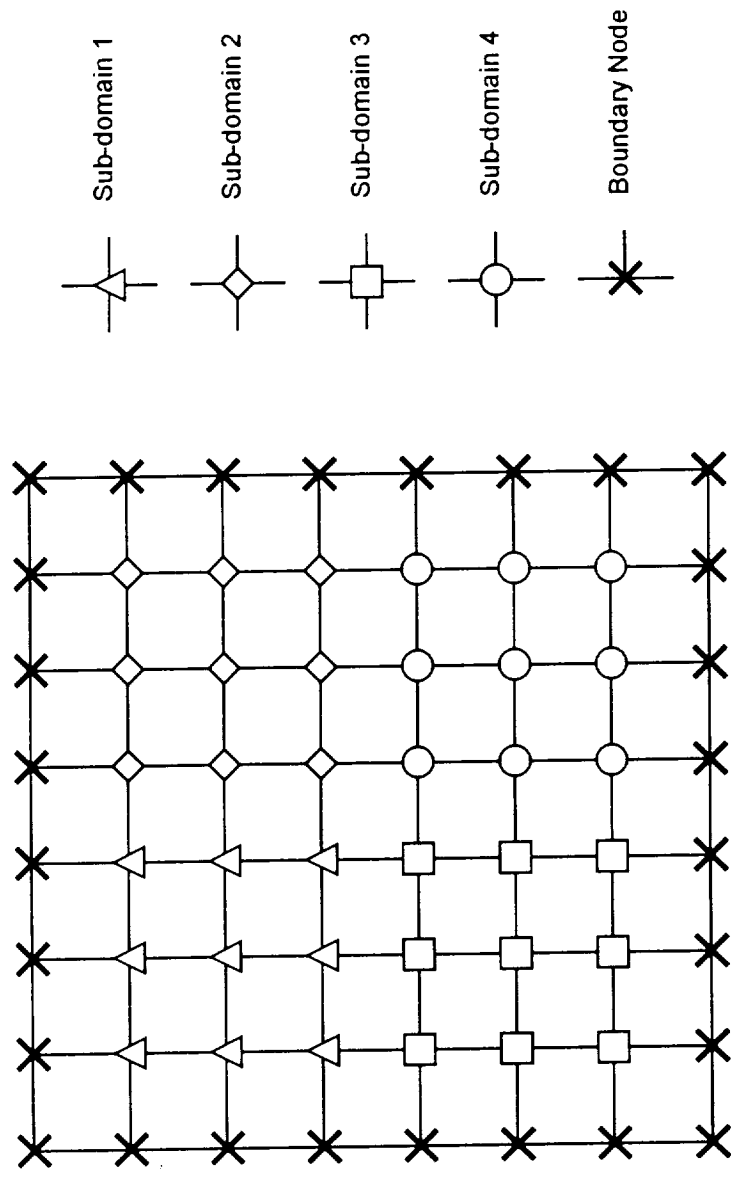
## Multi-threaded Programming

---

- On shared memory multi-processor machines, creating multiple threads is an ideal way to parallelize an application since individual threads may be assigned to separate CPU's by the OS.
- Balancing the computational load between multiple processors is critical to achieving a high degree of parallelism.
- A combined domain decomposition/multi-threaded approach is presented.

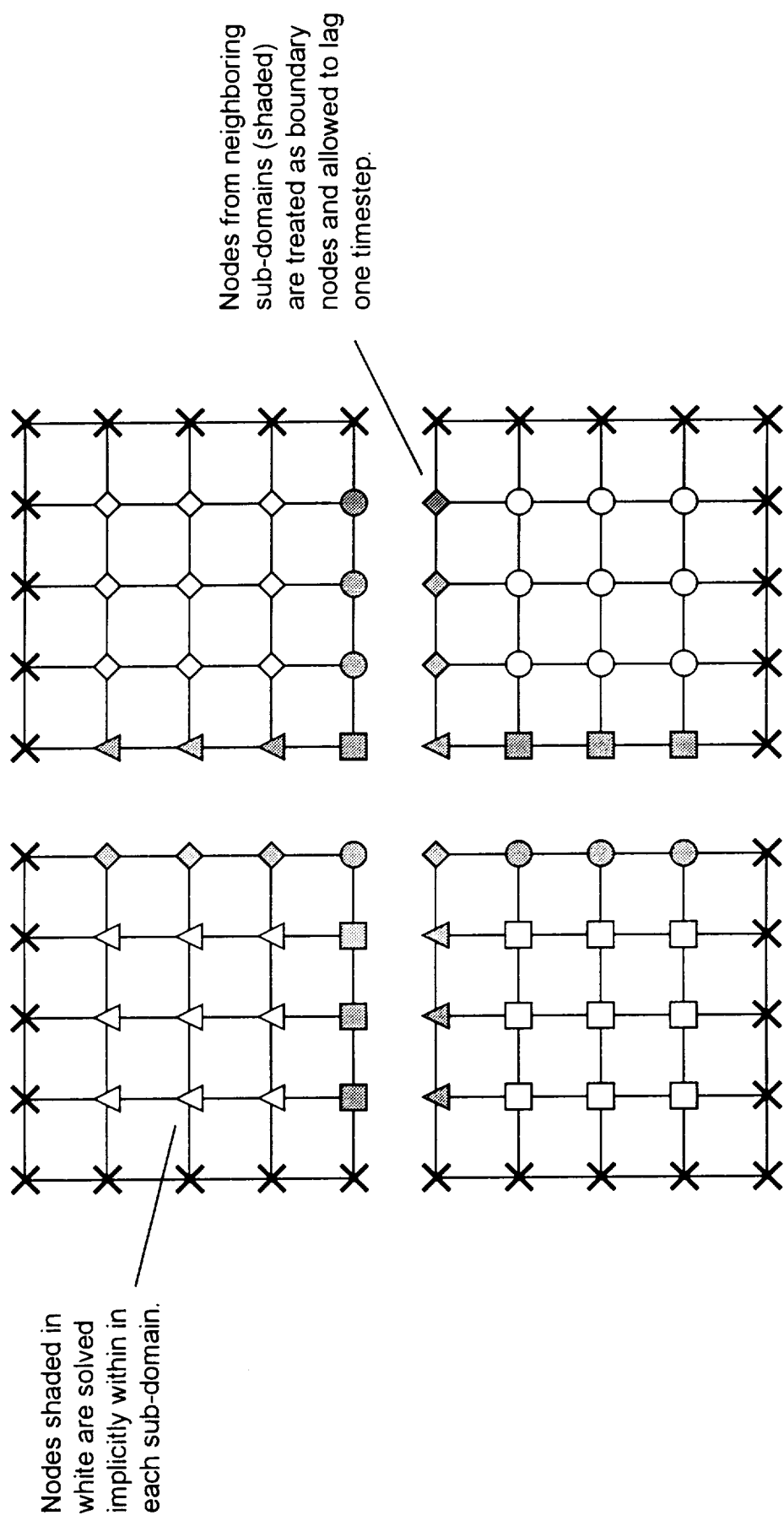
# Domain Decomposition

## Additive Schwarz Method with Overlapping Sub-domains



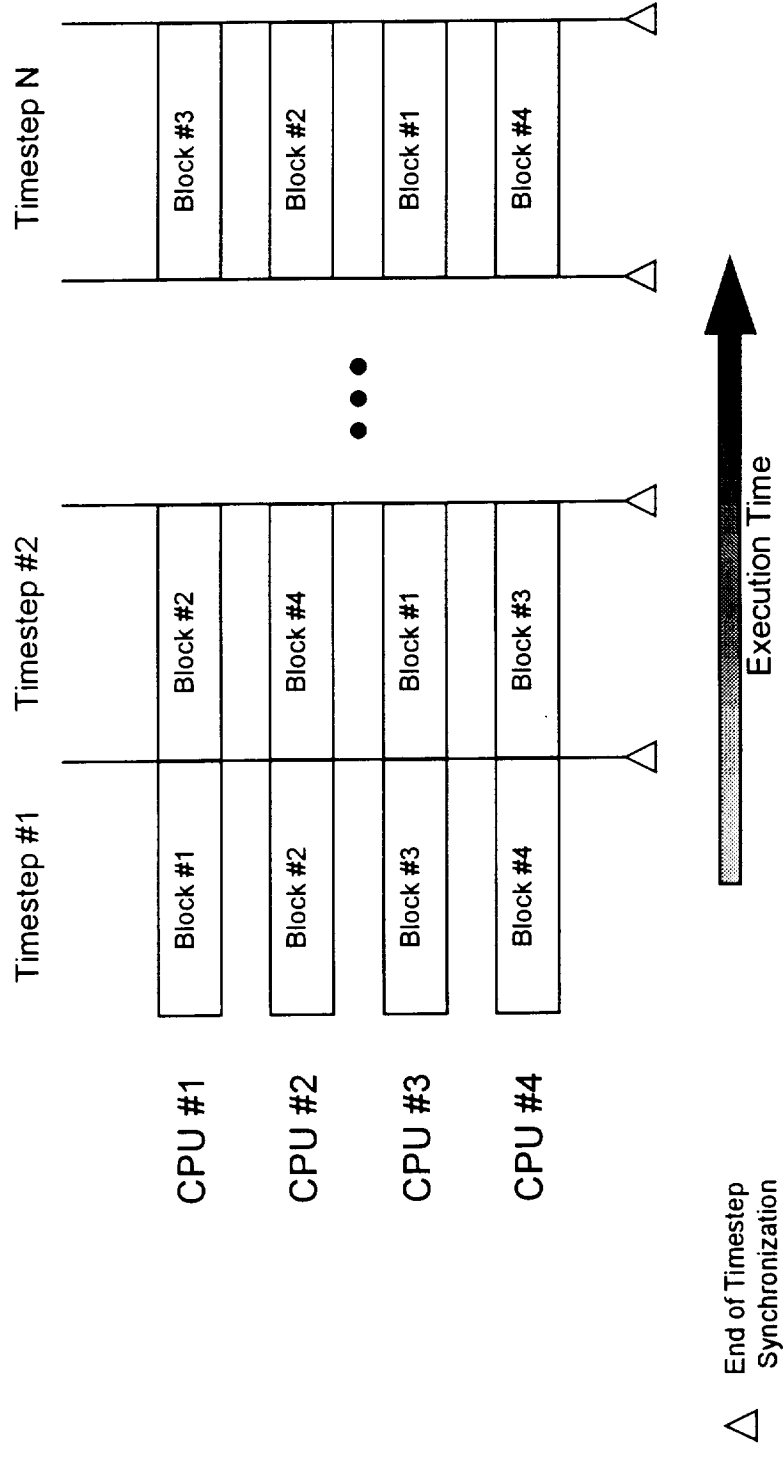
# Domain Decomposition

## Additive Schwarz Method with Overlapping Sub-domains

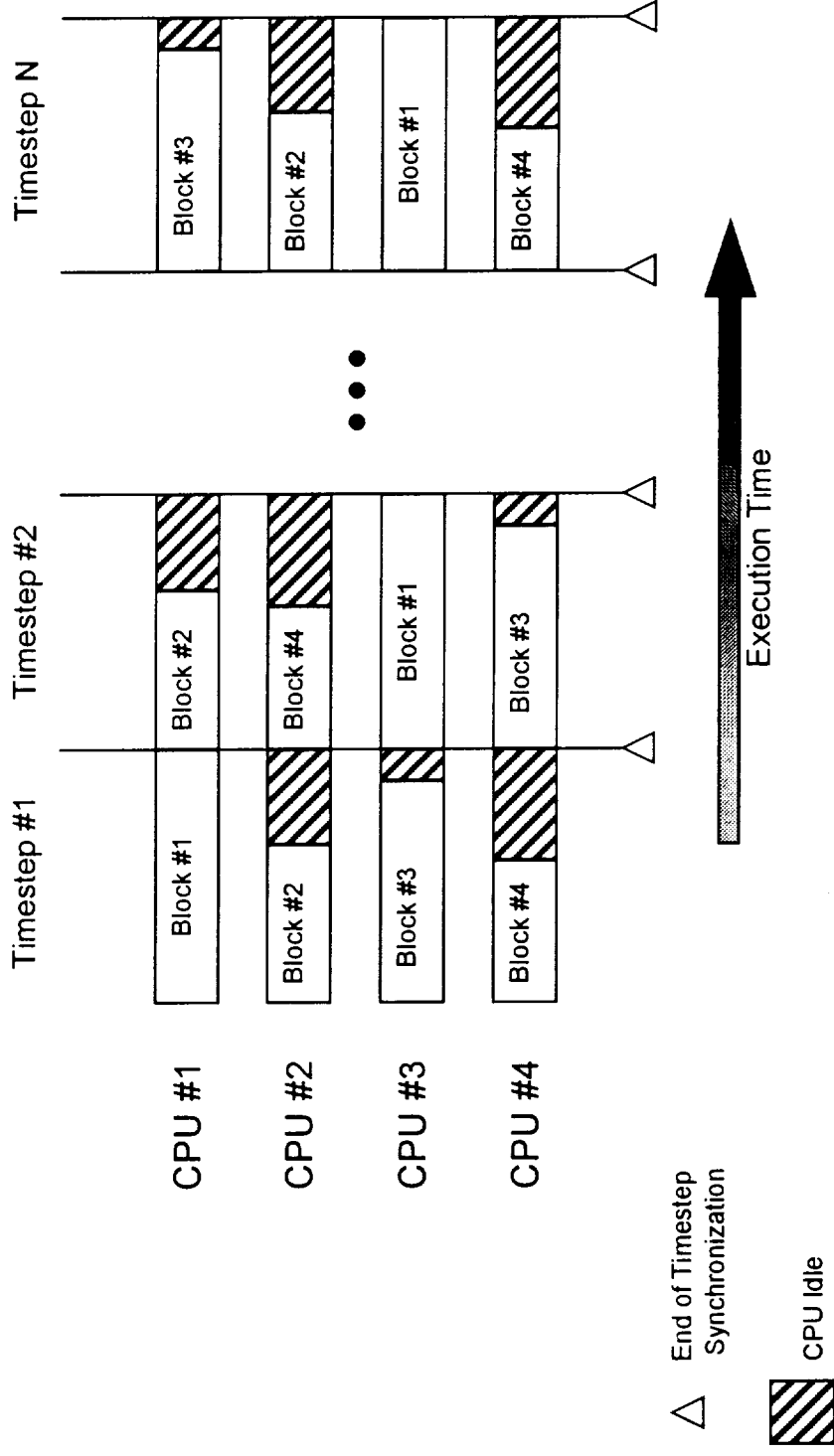


\*Each interior node is solved in an implicit fashion in exactly one sub-domain.

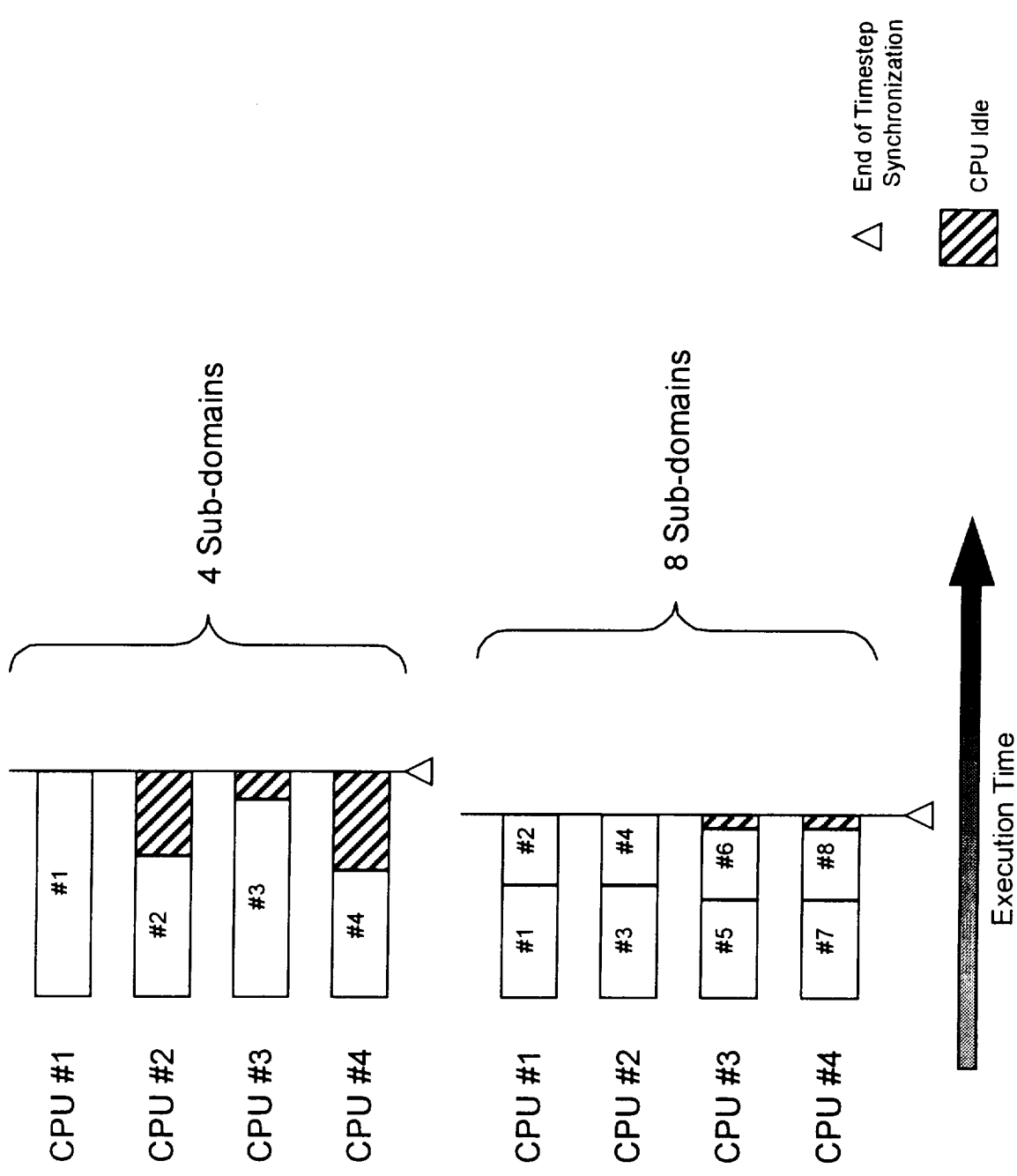
# Processor Load Balancing for the Ideal Case



# Processor Load Balancing in the “Real World”

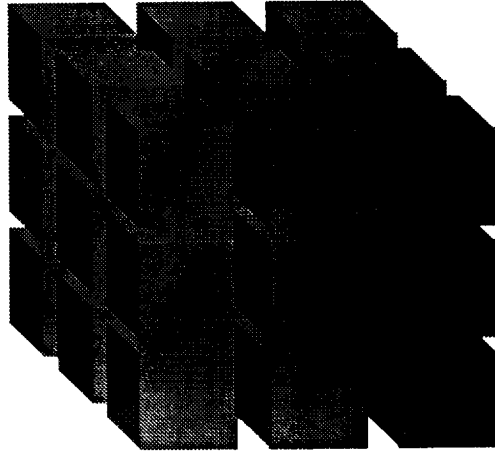


# Increasing the Number of Sub-domains Improves Load Balancing



# Multi-threaded Programming Implementation

---



*Decompose the domain*

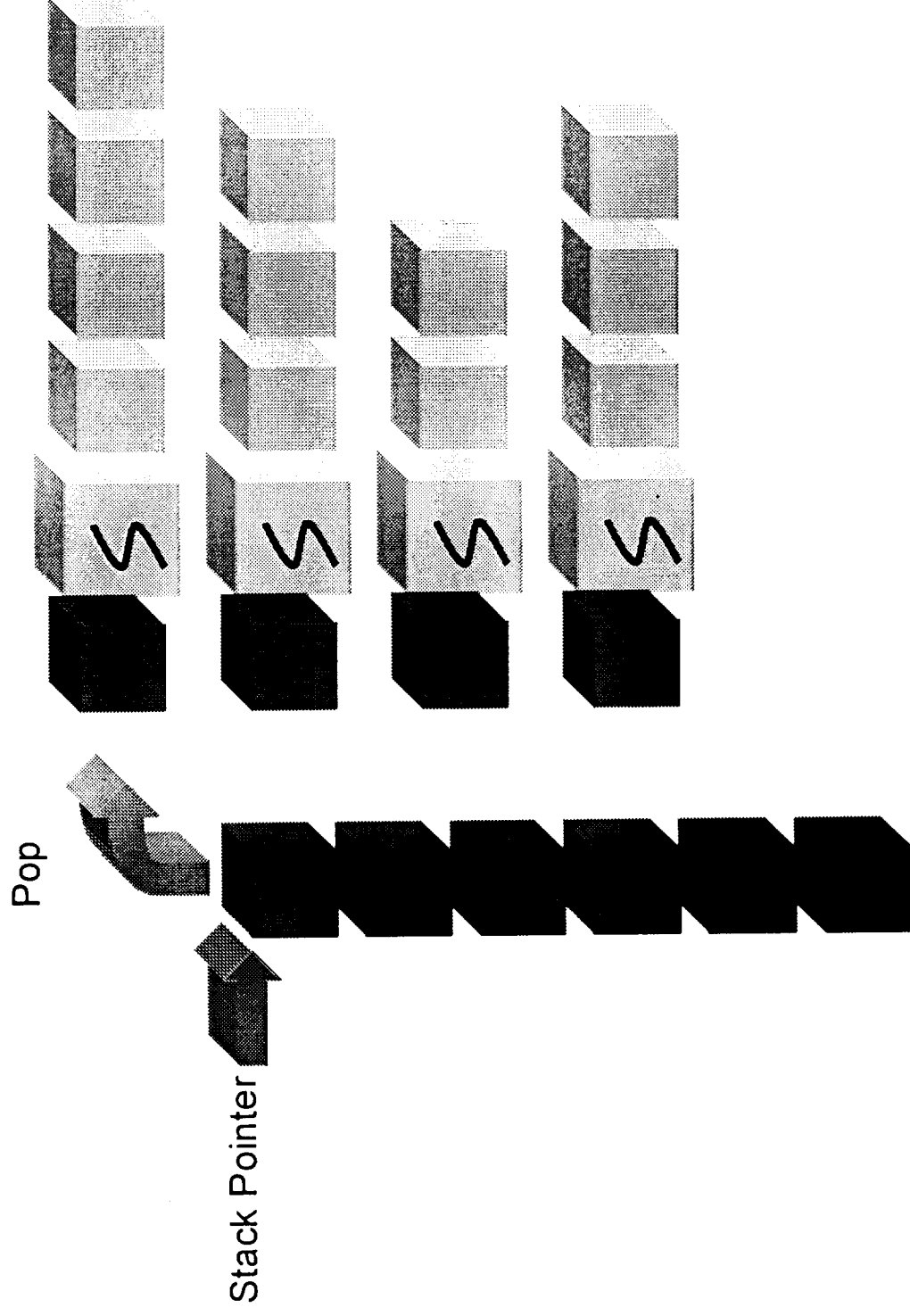


*Push each sub-domain onto a software stack*



# Multi-threaded Programming Implementation

---



*Spawn threads and execute until stack is exhausted*

# Computational Benchmarks

Threads	Grid	Decomposition	CPU Time (hours)	Elapsed Time (hours)	CPU Utilization	Speed-up	Processor	Number of Proc
1	55x41x31	4x4x4	5.05	5.05	100%	1.00	Pentium II	2
2	55x41x31	4x4x4	5.13	2.62	196%	1.93	Pentium II	2
1	55x41x31	4x4x4	4.69	4.72	99%	1.00	Alpha	4
2	55x41x31	4x4x4	5.19	2.66	195%	1.77	Alpha	4
4	55x41x31	4x4x4	5.30	1.42	373%	3.32	Alpha	4
6	55x41x31	4x4x4	5.30	1.40	378%	3.36	Alpha	4
8	55x41x31	4x4x4	5.16	1.37	377%	3.44	Alpha	4

## Conclusions and Future Plans

---

- Preliminary results are encouraging:
  - A more rigorous treatment of the chemical species generation terms may be necessary to relax timestep constraint.
- Incorporate the 28 reaction H<sub>2</sub>-Air chemical kinetic mechanism for comparison to the two step global reaction mechanism.
- Incorporate a Large Eddy Simulation or turbulence model to account for enhanced reaction rates due to turbulent mixing.
- Merge the Navier Stokes and species conservation solvers into one program. Migrate the application to a “truly” three dimensional benchmark.